CLAIMS

WE CLAIM:

1. A system, comprising:

a clock unit adapted to generate a first clock having a first clock unit interval and a first clock phase offset; and

a processing unit adapted to: add the first phase offset to a first timestamp of a plurality of timestamps to generate a first offset timestamp, integer divide the first offset timestamp by the first clock unit interval to identify a first integer, add the first phase offset to a second timestamp of the plurality of timestamps to generate a second offset timestamp, integer divide the second offset timestamp by the first clock unit interval to identify a second integer, and determine whether the first and second integers are equal.

2. The system of claim 1, further comprising a memory adapted to receive a plurality of data elements, each data element being associated with a timestamp, and to store the plurality of data elements and timestamps.

3. The system of claim 1, wherein the processing unit is further adapted to determine that the first clock is not a passing clock signal if the first and second integer are equal.

4. The system of claim 1, wherein the processing unit is further adapted to add the first phase offset to each timestamp of said plurality of timestamps to generate a plurality of offset timestamps, integer divide each of said plurality of offset timestamps by the first clock unit interval to identify a plurality of integers, and determine whether any of the plurality of integers are equal.

5. The system of claim 4, wherein the processing unit is further adapted to determine that the first clock is a passing clock signal if none of the plurality of integers are equal.

6. The system of claim 4, wherein the processing unit is further adapted to determine that the first clock is not a passing clock signal if any of the plurality of integers are equal.

7. The system of claim 6, wherein:

the clock unit is further adapted to generate a second clock having a second clock unit interval and a second clock phase offset; and

the processing unit is further adapted to: add the second phase offset to the first timestamp of the plurality of timestamps to generate a third offset timestamp, integer divide the third offset timestamp by the second clock unit interval to identify a third integer, add the second phase offset to the second timestamp of the plurality of timestamps to generate a forth offset timestamp, integer divide the forth offset timestamp by the second clock unit interval to identify a forth integer, and determine whether the third and forth integers are equal.

8. The system of claim 7, wherein the first and second clock unit intervals are equal.

9. The system of claim 8, wherein the first and second clock phase offsets are equal.

10. The system of claim 7, wherein the processing unit is further adapted to determine a range of passing phase offsets.

11. The system of claim 7, wherein the processing unit is further adapted to determine a range of passing unit intervals.

12. The system of claim 6, wherein:

the clock unit is further adapted to generate a second clock having the first clock unit interval and a second clock phase offset; and

the processing unit is further adapted to: add the second phase offset to the first timestamp of the plurality of timestamps to generate a third offset timestamp, integer divide the third offset timestamp by the first clock unit interval to identify a third integer, add the second phase offset to the second timestamp of the plurality of timestamps to generate a forth offset timestamp, integer divide the forth offset timestamp by the first clock unit interval to identify a forth integer, and determine whether the third and forth integers are equal.

13. The system of claim 1, further comprising a core logic unit.

14. The system of claim 1, further comprising a memory device.

15. A method, comprising:

generating a first clock having a first clock unit interval and a first clock phase offset;

adding the first phase offset to a first timestamp of a plurality of timestamps to generate a first offset timestamp;

integer dividing the first offset timestamp by the first clock unit interval to identify a first integer;

adding the first phase offset to a second timestamp of the plurality of timestamps to generate a second offset timestamp;

integer dividing the second offset timestamp by the first clock unit interval to identify a second integer; and

determining whether the first and second integers are equal.

16. The method of claim 15, further comprising:

receiving a plurality of data elements, each data element being associated with a timestamp; and

storing the plurality of data elements and timestamps.

17. The method of claim 15, further comprising:

determining that the first clock is not a passing clock signal if the first and second integers are equal.

18. The method of claim 15, further comprising:

adding the first phase offset to each timestamp of said plurality of timestamps to generate a plurality of offset timestamps;

integer dividing each of said plurality of offset timestamps by the first clock unit interval to identify a plurality of integers;

determining whether any of the plurality of integers are equal.

19. The method of claim 18, further comprising:

determining that the first clock is a passing clock signal if none of the plurality of integers are equal.

20. The method of claim 19, further comprising:

determining that the first clock is not a passing clock signal if any of the plurality of integers are equal.

21. The method of claim 20, further comprising:

generating a second clock having a second clock unit interval and a second clock phase offset;

adding the second phase offset to the first timestamp of the plurality of timestamps to generate a third offset timestamp;

integer dividing the third offset timestamp by the second clock unit interval to identify a third integer;

adding the second phase offset to the second timestamp of the plurality of timestamps to generate a forth offset timestamp;

integer dividing the forth offset timestamp by the second clock unit interval to identify a forth integer; and

determining whether the third and forth integers are equal.

22. The method of claim 21, wherein the first and second clock unit intervals are equal.

23. The method of claim 21, wherein the first and second clock phase offsets are equal.

24. The method of claim 21, further comprising determining a range of passing phase offsets.

25. The method of claim 21, further comprising determining a range of passing unit intervals.

26. The method of claim 20, further comprising:

generating a second clock having the first clock unit interval and a second clock phase offset;

adding the second phase offset to the first timestamp of the plurality of timestamps to generate a third offset timestamp;

integer dividing the third offset timestamp by the first clock unit interval to identify a third integer;

adding the second phase offset to the second timestamp of the plurality of timestamps to generate a forth offset timestamp;

integer dividing the forth offset timestamp by the first clock unit interval to identify a forth integer; and

determining whether the third and forth integers are equal.

27. The method of claim 15, wherein at least one of the steps of adding the first phase offset to a first timestamp of a plurality of timestamps to generate a first offset timestamp; integer dividing the first offset timestamp by the first clock unit interval to identify a first integer; adding the first phase offset to a second timestamp of the plurality of timestamps to generate a second offset timestamp; integer dividing the second offset timestamp by the first clock unit interval to identify a second integer; and determining whether the first and second integers are equal is performed using a core logic unit.

28. A computer-readable medium comprising computer-executable instructions which, when executed by a computer, cause the computer to perform the method of claim 15.

29. A method comprising:

calculating the difference in time between a first data timestamp and a first clock edge of a clock signal to identify a first jitter element;

calculating the difference in time between a second data timestamp and a second clock edge of a clock signal to identify a second jitter element;

analyzing the first and second jitter elements to identify jitter characteristics of data relative to a clock.

30. The method of claim 29, further comprising:

generating a first clock having a first clock unit interval and a first clock phase offset;

adding the first phase offset to a first timestamp of a plurality of timestamps to generate a first offset timestamp;

integer dividing the first offset timestamp by the first clock unit interval to identify a first integer;

adding the first phase offset to a second timestamp of the plurality of timestamps to generate a second offset timestamp;

integer dividing the second offset timestamp by the first clock unit interval to identify a second integer; and

determining whether the first and second integers are equal.